ESP8266

PWM 接口参考

Version 1.2

Espressif Systems IOT Team

http://bbs.espressif.com

免责申明和版权公告

本文中的信息，包括供参考的 URL 地址，如有变更，恕不另行通知。

文档“按现状”提供，不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档不负任何责任，包括使用本文档内信息产生的侵犯任何专利权行为的责任。本文档在此未以禁止反言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。

Wi-Fi 联盟成员标志归 Wi-Fi 联盟所有。

文中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。

Table of Contents

1. 概述

1.1. 特性描述

ESP8266 系统的 PWM（Pulse Width Modulation）由 FRC1 在软件上实现，可实现同频率、不同占空比的多路 PWM，可用来控制彩灯、蜂鸣器和电机等设备。

> 说明：
> FRC1 是一个 23 bit 的硬件定时器。

PWM 的特性如下所示。

- 使用 NMI（Non Maskable Interrupt）中断，更加精确。
- 可扩展最多 8 路 PWM 信号。
- >14 bit 分辨率，最小分辨率 45 ns。
- 无需配置寄存器，调用函数接口即可完成配置。

> 注意：
> - PWM 驱动接口不能跟硬件定时器 (hw_timer) 接口函数同时使用，因为二者共用同一个硬件定时器。
> - 如需使用 PWM 驱动，请勿调用 wifi_set_sleep_type(LIGT_SLEEP); 将自动睡眠模式设置为 Light Sleep。因为 Light Sleep 在睡眠期间会停 CPU，停 CPU 期间不能响应 NMI 中断。
> - 如需进入 Deep Sleep，请先将 PWM 关闭，再进行休眠。

1.2. 实现方式

ESP8266 系统提供了一种经过优化的软件算法，通过在 FRC1 定时器上挂载 NMI，实现在 GPIO（General Purpose Input Output） 端口输出多组 PWM 信号。

PWM 的时钟源由高速系统时钟提供，其频率高达 80 MHz。PWM 通过预分频器将时钟源16分频，其输入时钟频率为 5MHz。PWM 通过 FRC1 来产生粗调定时，结合高速系统时钟的微调，可将分辨率提高到 45 ns。

> 说明：
> NMI 拥有最高中断优先级，可以保证 PWM 输出波形的准确度。

1.3. 配置说明

- 在定时中断内，为尽快退出程序只在每次 PWM 周期开始时载入下一个周期的定时参数。

- 设置完各个通道的占空比后，系统会调用 `pwm_start()` 函数来计算定时周期。在此之前系统会进行保护操作，即保存当前的各通道参数，并清除计算完成标志， PWM 周期到来会使用保存后的参数。
- PWM 周期中断后会使用新的参数，因此计算完成后需要设置标志位。这样在实现占空比渐变（如控制 RGB 彩灯）的过程中，能保证颜色平滑过渡。
- 可在 user_light.h 中配置采用的 GPIO。SDK 代码示例使用 5 路 PWM，实际可以自行扩展，最多扩展至8路PWM，具体参见"第 3 章 自定义通道"。最小分辨率 45 ns，频率在 1KHz 时，占空比最小可以达到 1/22222。

1.4. 参数说明

- 最小分辨率：45 ns (近似对应于硬件 PWM 的输入时钟频率为 22.72 MHz) : >14 bit PWM @ 1 KHz
- PWM 周期：1000 μs（1 KHz） ~ 10000 μs（100 Hz）

2. pwm.h 详解

2.1. 代码示例

```
#ifndef __PWM_H__
#define __PWM_H__
#define PWM_CHANNEL_NUM_MAX 8                   //最多 8 路PWM。
struct pwm_single_param {                       //定义单个 PWM 通道参数结构体。
    uint16 gpio_set;                            //需要置位的 GPIO。
    uint16 gpio_clear;                          //需要清零的 GPIO。
    uint32 h_time;                              //需要写入 FRC1_LOAD 寄存器的计数值。
};
struct pwm_param {                              //定义 PWM 参数结构体。
    Uint32 period;                              //PWM 周期。
    Uint32 freq;                                //PWM 频率。
    uint32 duty[PWM_CHANNEL_NUM_MAX];           //PWM 占空比。
};
void pwm_init(uint32 period, uint32 *duty,uint32 pwm_channel_num,uint32
(*pin_info_list)[3]);
void pwm_start(void);
void pwm_set_duty(uint32 duty, uint8 channel);
uint32 pwm_get_duty(uint8 channel);
void pwm_set_freq(uint32 period);
uint32 pwm_get_freq(void);
```

2.2. 接口说明

2.2.1. pwm_init

名称	pwm_init
含义	PWM 初始化。

代码示例	`pwm_init (uint32 freq, uint32 *duty, uint32 pwm_channel_num,uint32 (*pin_info_list)[3]);`
描述	PWM GPIO，参数和定时器初始化。

参数	• uint32 freq：PWM 的周期。
	• uint32 *duty：各通道占空比参数。
	• uint32 pwm_channel_num：PWM 通道数。
	• uint32 (*pin_info_list)[3]：PWM 各通道的 GPIO 硬件参数，该参数是一个 n x 3 的数组指针。数组中定义了 GPIO 的寄存器，对应 PIN 脚的 IO 复用值，和 GPIO 对应的序号。
	例如：初始化一个 3 通道的 PWM。
	`uint32 io_info[][3] =`
	`{{PWM_0_OUT_IO_MUX,PWM_0_OUT_IO_FUNC,PWM_0_OUT_IO_NUM},`
	`{PWM_1_OUT_IO_MUX,PWM_1_OUT_IO_FUNC,PWM_1_OUT_IO_NUM},`
	`{PWM_2_OUT_IO_MUX,PWM_2_OUT_IO_FUNC,PWM_2_OUT_IO_NUM}};`
	`pwm_init(light_param.pwm_period,light_param.pwm_duty,3,io_info);`

调用	系统初始化时调用。目前只能调用一次。
返回值	无

2.2.2. pwm_set_period

名称	pwm_set_period
含义	设置 PWM 周期。
代码示例	`pwm_set_period (uint32 period)`
描述	设置 PWM 周期，单位 μs。 例如：1KHz PWM，参数为 1000 μs。
参数说明	uint32 period：PWM 周期。
调用	设置完成后需要调用 `pwm_start()` 才起作用。
返回值	无

2.2.3. pwm_set_duty

名称	pwm_set_duty
含义	设置 PWM 某个通道信号的占空比。
代码示例	`pwm_set_duty (uint32 duty, uint8 channel)`
描述	设置 PWM 占空比。设置各路 PWM 信号高电平所占的时间，duty 的范围随PWM周期改变。最大值为：period*1000/45 (以1kHz为例：duty 范围是 0～22222)。

参数说明	• uint32 duty：设置高电平时间参数，占空比的值为 (duty*45)/ (period*1000)。 • uint8 channel：当前要设置的 PWM 通道，在 PWM_CHANNEL 定义的范围内。
调用	设置完成后需要调用 `pwm_start()` 才起作用。
返回值	无

2.2.4. pwm_get_period

名称	pwm_get_period
描述	获取当前 PWM 周期。
代码示例	`pwm_get_period (void)`
参数说明	无
返回值	PWM 周期，单位 μs。

2.2.5. pwm_get_duty

名称	pwm_get_duty
描述	获取对应 channel 的当前 PWM 信号的 duty 参数。
代码示例	`pwm_get_duty (uint8 channel)`
参数说明	uint8 channel：当前要获取的 PWM 通道，在 PWM_CHANNEL 定义的范围内。
调用	设置完成后需要调用 `pwm_start()` 才起作用。
返回值	channel 对应的通道的占空比，占空比的值为 (duty*45)/ (period*1000)。

2.2.6. pwm_start

名称	pwm_start
描述	PWM 更新参数。
代码示例	`pwm_start (void)`
参数说明	无
调用	PWM 相关参数设置完成后，需要调用 `pwm_start()` 才起作用。
返回值	无

3. 自定义通道

用户还可以增加 PWM 通道，如需增加 GPIO4 为 PWM 输出的第四通道，设置步骤如下所示。

1. 修改初始化参数。

```
uint32 io_info[][3]={
    {PWM_0_OUT_IO_MUX,PWM_0_OUT_IO_FUNC,PWM_0_OUT_IO_NUM},
    {PWM_1_OUT_IO_MUX,PWM_1_OUT_IO_FUNC,PWM_1_OUT_IO_NUM},
    {PWM_2_OUT_IO_MUX,PWM_2_OUT_IO_FUNC,PWM_2_OUT_IO_NUM},
    {PWM_3_OUT_IO_MUX,PWM_3_OUT_IO_FUNC,PWM_3_OUT_IO_NUM},
    {PWM_4_OUT_IO_MUX,PWM_4_OUT_IO_FUNC,PWM_4_OUT_IO_NUM},
    };
pwm_init(light_param.pwm_period, light_param.pwm_duty, PWM_CHANNEL,io_info);
```

2. 修改 user_light.h 文件。

```
#define PWM_0_OUT_IO_MUX PERIPHS_IO_MUX_MTDI_U
#define PWM_0_OUT_IO_NUM 12
#define PWM_0_OUT_IO_FUNC FUNC_GPIO12
#define PWM_1_OUT_IO_MUX PERIPHS_IO_MUX_MTDO_U
#define PWM_1_OUT_IO_NUM 15
#define PWM_1_OUT_IO_FUNC FUNC_GPIO15
#define PWM_2_OUT_IO_MUX PERIPHS_IO_MUX_MTCK_U
#define PWM_2_OUT_IO_NUM 13
#define PWM_2_OUT_IO_FUN CFUNC_GPIO13
#define PWM_3_OUT_IO_MUX PERIPHS_IO_MUX_GPIO4_U
#define PWM_3_OUT_IO_NUM 4
#define PWM_3_OUT_IO_FUNC FUNC_GPIO4
#define PWM_4_OUT_IO_MUX PERIPHS_IO_MUX_GPIO5_U
#define PWM_4_OUT_IO_NUM 5
#define PWM_4_OUT_IO_FUNC FUNC_GPIO5
#define PWM_CHANNEL 5
```

——结束